Ruby - Feature #7394

Enumerable#find ifnone parameter could be non-callable

11/19/2012 12:36 PM - zzak (zzak _)

Status:	Assigned	
Priority:	Normal	
Assignee:	nobu (Nobuyoshi Nakada)	
Target version:		

Description

from github:
https://github.com/ruby/ruby/pull/186

In trunk the Enumerable#find method ifnone parameter has to be callable or nil. I found that sometimes I want to return a simple value without wrapping it in a proc. This pull request adds support for non-callable defaults when no items match.

```
a = [1, 2, 3]
```

The current behavior

```
a.find(proc { :foo }) { |x| x > 3 } #=> :foo
```

With patch

```
a.find(0) { |x| x > 3 } #=> 0
```

History

#1 - 11/24/2012 11:02 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Target version changed from 2.0.0 to 2.6

Zachary Scott, please don't add 2.0.0 feature ticket unless I approve. The 2.0.0 feature deadline was passed.

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 11/24/2012 11:09 AM - mame (Yusuke Endoh)

Oh, I didn't realized that this ticket was from github pull reqeust.
Thank you for your importing work!
But, the fact remains that this proposal was not accepted by the 2.0.0 deadline. Sorry.

It is unfortunate that people misunderstands that github pullreq is the right way to request a feature to Ruby.
Is it impossible to stop (or automatically reject) pullreq?

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 11/24/2012 12:16 PM - zzak (zzak _)

This was during my import of patches from github to redmine.

You can turn off pull requests on github, maybe start a new thread to discuss that.

#4 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#5 - 02/13/2019 10:06 PM - ioquatix (Samuel Williams)

Can we merge this?

#6 - 02/13/2019 10:08 PM - ioquatix (Samuel Williams)

Just because it might cause some surprise, perhaps we can use keyword argument for this.

e.g.

```
a = [1, 2, 3]
a.find(else: 0) { |x| x > 3 } #=> 0
```

I kind of prefer original suggestion, but I can imagine if passing an object that responds to #call it might have unexpected behaviour.

#7 - 02/14/2019 04:44 AM - nobu (Nobuyoshi Nakada)

- Description updated

Currently, it is not able to distinguish from the case a hash is given as an ordinal argument.
So there still is a possibility to break a compatibility.

#8 - 02/10/2021 07:48 AM - baweaver (Brandon Weaver)

I rather like @ioquatix (Samuel Williams) idea here, and was considering making a similar ticket to suggest that ifnone does not make much sense when compared to other Ruby APIs.

Would we still consider merging this?

#9 - 02/10/2021 07:53 AM - baweaver (Brandon Weaver)

A second argument, however, could be made that find_index provides a different API:

```
(1..100).find_index(50)
```

...in which we have a similar API for predicate methods (any?),grep, and other Enumerable methods.

I could see cases for both, but the original proposed in this thread would make the most sense given the current API to avoid breakage.

#10 - 02/10/2021 09:32 AM - zverok (Victor Shepelev)

What's the point of the "default value" as compared to just find { ... } || default?

- Would it be more performant? (I believe no)
- Would it read more naturally? I believe no, the statement reads "find (and if not found, use that) by this condition..."
- Would it be readable at all?.. I believe barely, actually find(10) { condition } can be misunderstood as "find, starting from index 10", and something like find(:delete) { condition } can be read as some local DSL for "find&delete". Keyword argument will improve it, of course, but looks unlike any other core API, and still reads "find, else default, by this condition...".

But the question for me is still: why is it necessary at all? what exactly it achieves?

Files

enumerable_find_noncallable.patch	3.45 KB	11/19/2012	zzak (zzak _)